CS 61B Spring 2025

Graphs II, Tries

Exam-Level 10: April 07, 2025

1 Multiple MSTs

Recall a graph can have multiple MSTs if there are multiple spanning trees of minimum weight.

(a) For each subpart below, select the correct option and justify your answer. If you select "never" or "always," provide a short explanation. If you select "sometimes," provide two graphs that fulfill the given properties.

1. If **some** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

2. If **all** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

(b) Suppose we have a connected, undirected graph (G) with (N) vertices and (N) edges, where all the **edge weights are identical**. Find the maximum and minimum number of MSTs in (G) and explain your reasoning.

Minimum:

Maximum:

Justification:

(c) It is possible that Prim's and Kruskal's find different MSTs on the same graph G (as an added exercise, construct a graph where this is the case!).

Given any graph G with integer edge weights, modify the edge weights of G to *ensure* that

(1) Prim's and Kruskal's will output the same results, and

(2) the output edges still form a MST correctly in the original graph.

You may not modify Prim's or Kruskal's, and you may not add or remove any nodes/edges.

Hint: Look at subpart 1 of part (a).

2 Topological Sorting for Cats

The big brain cat, Duncan, is currently studying topological sorts! However, he has a variety of curiosities that he wishes to satisfy.

(a) Describe at a high level in plain English how to perform a topological sort using an algorithm we already know (hint: it involves DFS), and provide the time complexity.

(b) Duncan came up with another way to possibly do topological sorts, and he wants you to check him on its correctness and tell him if it is more efficient than our current way! Let's derive the algorithm.

1. First, provide a logical reasoning for the following claim (or a proof!): Every DAG has at least one source node, and at least one sink node.

2. Next, describe an algorithm (in English or in pseudocode) for finding all of the source nodes in a graph.

3. Now, observe that if we remove all of the source nodes from a DAG, we are guaranteed to have at least one new source node. Inspired by this fact, and using the previous parts, come up with an algorithm to topological sort. Is it more efficient than what we already have?

4. Dawn, the garbage collector from Brentwood is allergic to cats. He wanted to trick Duncan and created a DAG object, but it actually represents a graph with a cycle! How can you modify the algorithm above to detect whether the graph has a cycle?

3 A Wordsearch

Given an N by N wordsearch and N words, devise an algorithm (using pseudocode or describe it in plain English) to solve the wordsearch in $O(N^3)$. For simplicity, assume no word is contained within another, i.e., if the word **"bear"** is given, **"be"** wouldn't also be given.

If you are unfamiliar with wordsearches or want to gain some wordsearch solving intuition, see below for an example wordsearch.

Example Wordsearch:

S F T A T R D V R G K E V I N
A T S K L N X F I H D P X H Z
C N E D X A J Z G U N A I R U
J Y I L C V A N E S S A V P O
A B A R L K F J Q U S Y H I C
V Z U I U A T Q K D A A G R D
F S P E I D S T A A S N M Y N
W C T T S S H Q T W H N G A U
S I W H P E A A E N L I T N V
T Y I A G L D B R K E Y T Y K
A L N N J A J G E T Y A P E A
C X F I K I M U S L I T Y P R
E M U A M N T M A Z X A H U E
Y R A E K E Y W I K O Y O P N
R B C A Q J V Q I A C R O E F

Anirudh	Anniyat	Vanessa	Ryan
Ashley	Elaine	Isabel	
David	Stella	Karen	
Ethan	Kevin	Teresa	
Stacey	Dawn		

Hint: Add the words to a **Trie**, and you may find the `longestPrefixOf` operation helpful. Recall that `longestPrefixOf` accepts a `String key` and returns the longest prefix of `key` that exists in the **Trie**, or `null` if no prefix exists.